

Deploying PeopleSoft with Stingray Traffic Manager

Using Stingray Traffic Manager with Oracle's
PeopleSoft Enterprise Suite

Table of Contents

Introduction	2
Why use Stingray Traffic Manager to manage PeopleSoft?	3
Reliability	3
Acceleration	3
Security	3
Management	4
Deploying Stingray Traffic Manager with PeopleSoft	4
Prerequisites	4
Software Versions	4
Example Architecture	5
Basic Configuration	5
Creating the Traffic IP Group	6
Creating Pools	6
Creating the Virtual Server	7
Starting the virtual server	8
Passing the Client IP address to WebLogic Server	8
Enabling Session Persistence	9
Monitor Application Cookies	10
Transparent Session Affinity	10
J2EE JSESSIONID cookies/URL	11
Load Balancing Algorithms	11
SSL Offload	11
Uploading your SSL private key and certificate to Stingray Traffic Manager	11
Configuring SSL Decryption	12
Notifying WebLogic that the connection was encrypted	12
Running both HTTP and HTTPS versions of a website	12
Traffic Routing (Separation of static and dynamic content)	12
About Riverbed	13

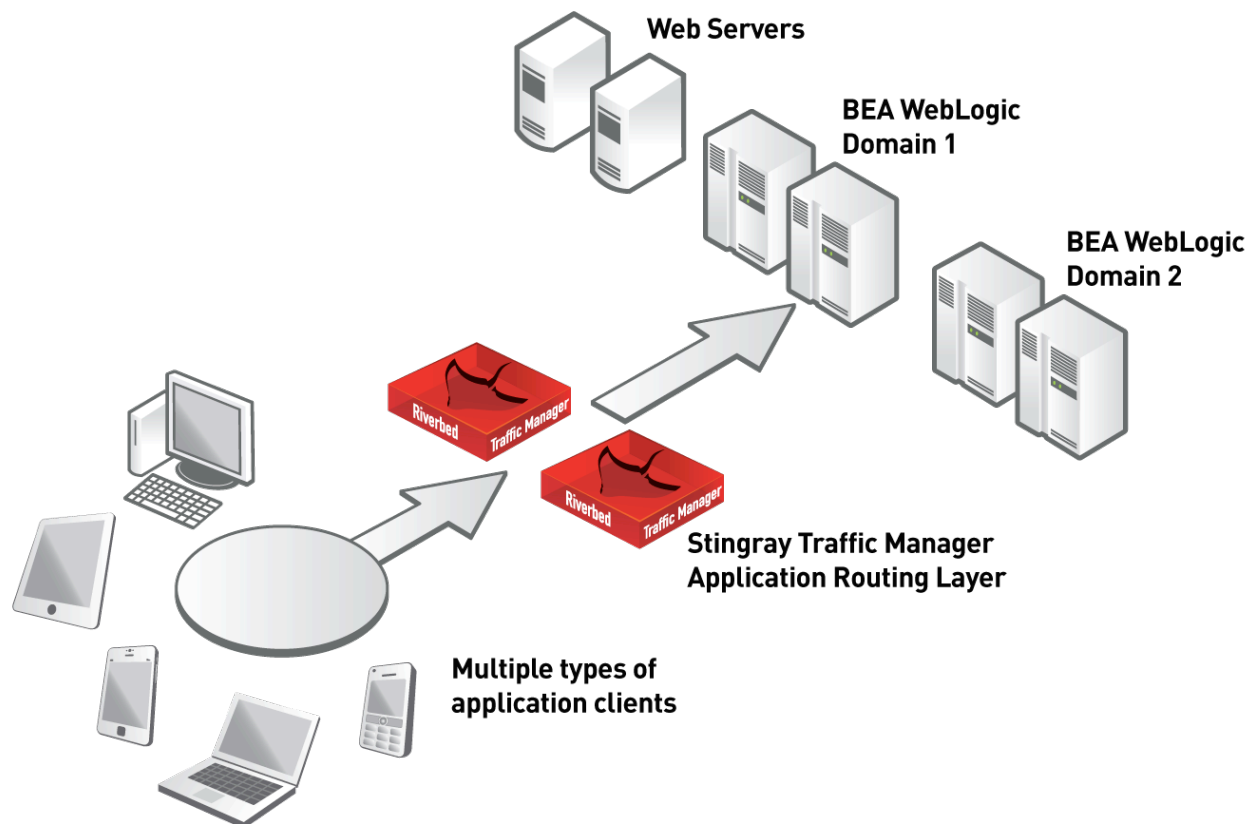
Introduction

Oracle's PeopleSoft Enterprise applications are designed to address the most complex business requirements. The PeopleSoft Enterprise suite is a comprehensive set of enterprise applications covering a range of business needs. Riverbed® Stingray™ Traffic Manager is able to accelerate web-based PeopleSoft applications, making them faster, more reliable, more secure and easier to manage.

Oracle's PeopleSoft uses either BEA WebLogic or IBM WebSphere as its Application Server. We describe here the process for integrating Stingray Traffic Manager with PeopleSoft using WebLogic.



Stingray Traffic Manager can improve the reliability of WebLogic applications such as PeopleSoft, double their performance (and hence double ROI), protect them from direct attacks and flash floods, and dramatically reduce the operational costs in managing servers and application upgrades.



Why use Stingray Traffic Manager to manage PeopleSoft?

Reliability

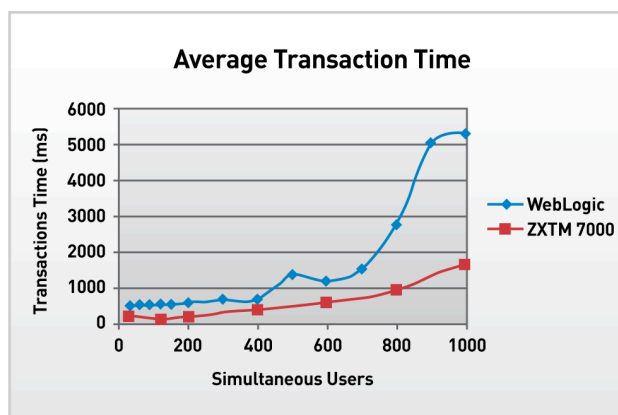
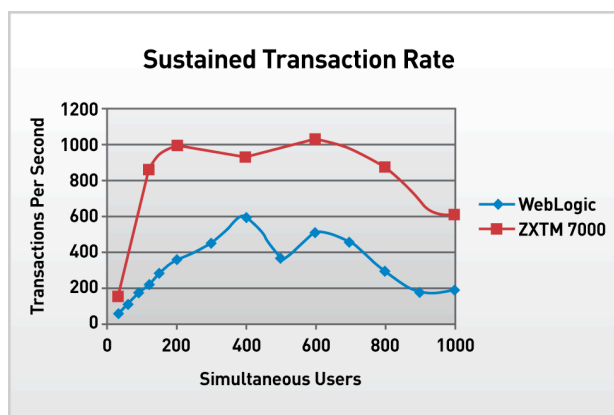
Stingray Traffic Manager provides a single point-of-entry to a PeopleSoft deployment served from a cluster of WebLogic servers. With a combination of passive and active health monitoring and performance measurements, Stingray Traffic Manager can route incoming requests to the fastest-responding servers and avoid servers that are failed or underperforming.

Request rate shaping ensures that your WebLogic servers can never be overloaded with requests, and advanced Session Persistence reliably pins users' sessions to individual servers for maximum reliability.

Acceleration

Stingray Traffic Manager has been proven to¹:

- **As much as double the transaction rate** that can be achieved from WebLogic, with no additional software or server tuning.
- **Double the response speed**, under sustained load from multiple clients.
- Provide over 15-times the SSL performance.
- **Eliminate all errors** for all but the most demanding tests.



Security

Total server isolation, request and response scrubbing, request validation and request rate shaping help to place your PeopleSoft servers in as secure an environment as possible, protecting them from direct attacks from internet clients, invalid or malformed HTTP requests and malicious or incidental flash floods that would severely impact the levels of service you provide.

Management

Stingray Traffic Manager's advanced request routing and manipulation, driven by a fully programmable TrafficScript rules engine, gives you full flexibility to manage both your network traffic and server resources:

- **Route traffic over multiple domains**, draining traffic from one domain to another as applications are upgraded and user sessions complete.
- **Load-balance over PeopleSoft servers and other servers** to optimize resource utilization and eliminate unnecessary load on your application.

¹ Accelerating BEA WebLogic with Stingray Traffic Manager – an independent report by BroadBand Testing. <http://www.riverbed.com/>

- Drain traffic from individual PeopleSoft servers, so that they can be taken out of service without any interruption to user's sessions.

Deploying Stingray Traffic Manager with PeopleSoft

Prerequisites

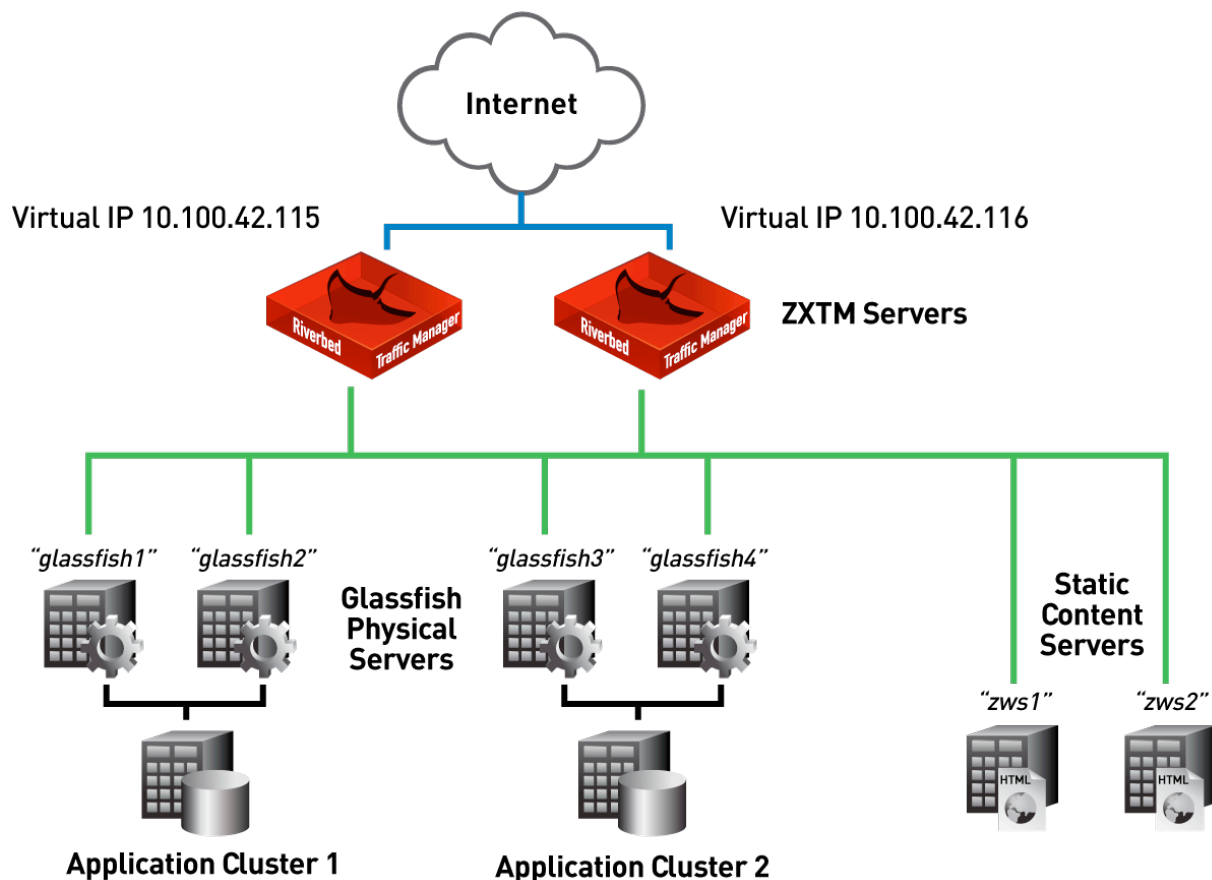
Software Versions

- Stingray Traffic Manager: Stingray Traffic Manager version 4.0 or later is required
- PeopleSoft Enterprise, using BEA WebLogic Server: Stingray Traffic Manager may be used to manage traffic to all versions of WebLogic Server. This document refers particularly to versions 8.x and 9.x.

It is assumed that the reader has followed the instructions in the Stingray Traffic Manager Getting Started Guide to install Stingray Traffic Manager on one or more machines in front of a cluster of WebLogic Servers running a PeopleSoft service. For installation and configuration of BEA WebLogic Server, please refer to the WebLogic Server documentation available from <http://edocs.bea.com/> for your version of BEA WebLogic. For installation and configuration of PeopleSoft, please refer the the accompanying documentation.

Example Architecture

The diagram below shows a typical deployment of Stingray Traffic Manager with PeopleSoft. A pair of Stingray Traffic Manager servers have been installed in front of two WebLogic application server clusters and a pair of web servers that serve static content.



The examples given in the rest of this document will refer to this diagram, and will use the IP addresses specified in it.

The Stingray Traffic Manager servers are shown operating in active-active mode, using a pair of Traffic IP addresses (x.x.x.10 and

x.x.x.20). The DNS for the sites hosted by this architecture would list two A records (one for x.x.x.10 and one for x.x.x.20) for each web site hostname.

Stingray Traffic Manager may also be configured to operate in active-passive mode; only a single IP address (x.x.x.10, for example) would be required for each hostname.

Although two application server clusters have been shown here in order to demonstrate request routing, Stingray Traffic Manager can be deployed in front of a single cluster. Similarly, a separate web-server cluster for static content is not essential, although it is recommended in many deployments since it reduces the load on the application servers.

Basic Configuration

To prepare Stingray Traffic Manager for load balancing traffic across a PeopleSoft cluster, you must perform the following tasks:

1. Create a Traffic IP group
2. Create a pool for each WebLogic cluster, containing all the servers in the cluster
3. Create a virtual server, bound to the traffic IP group, that will listen on the appropriate port and send traffic to the pool
4. Start the virtual server

Begin by logging in to the Stingray Traffic Manager administration server.

Creating the Traffic IP Group

Go to **Services > Traffic IP Group**, and create a new traffic IP group, containing the external IP address(es) to which the hostnames of your web sites resolve. The example group is named "weblogic_example".

The screenshot shows a web form titled "Create a new Traffic IP Group". It has several fields: "Name" with the value "weblogic_example", "Traffic Managers" with a dropdown set to "Traffic Manager", and "IP Addresses" with the value "x.x.x.10 x.x.x.20". Below the "Traffic Managers" dropdown, there is a table-like structure showing a server icon, the hostname "hanjague.home.menavaur.org", the IP "10.59.1.1", and a checked checkbox in the "Add" column. At the bottom of the form is a button labeled "Create Traffic IP Group".

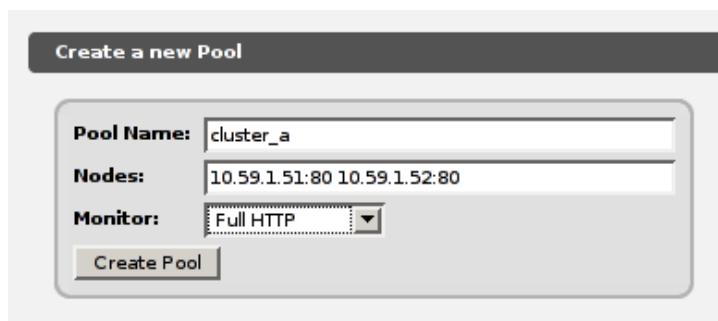
Creating Pools

To create a new pool for your WebLogic servers, go to **Services > Pool**:

Enter a name for the pool, and a space separated list of nodes. Each node is defined as "hostname:port" where the hostname is the hostname or IP address of the node, and the port is the number of the TCP/UDP port to which traffic should be sent (e.g. "10.10.0.56:80" or "apps-1.localdomain:443").

Set the monitor to "Full HTTP".

Click "Create Pool"



Create a new Pool

Pool Name: cluster_a

Nodes: 10.59.1.51:80 10.59.1.52:80

Monitor: Full HTTP

Create Pool

The screenshot shows this being done for the two nodes in Apps Server Cluster A, which are listening on port 80.

Repeat this step for each pool of application servers and web servers you wish to load balance. With the example architecture, two further pools would be configured as follows:

```
Pool Name: cluster_b
Nodes : 10.59.1.61:80 10.59.1.62:80
Monitor : Full HTTP

Pool Name: web_servers
Nodes : 10.59.1.31:80 10.59.1.32:80
Monitor : Full HTTP
```

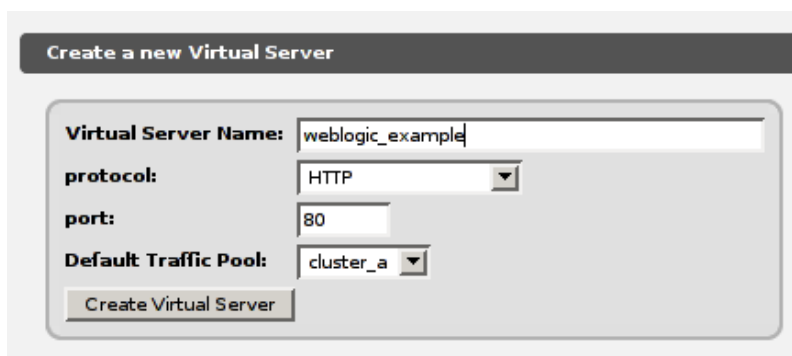
Creating the Virtual Server

Go to **Services > Virtual Servers**:

In the **Create a New Virtual Server** dialog, choose a name for the virtual server, set the protocol to HTTP, and the port to 80 (this is the port that Stingray Traffic Manager will listen on for incoming traffic).

Choose a default pool to send traffic to (in the example, this has been set to the cluster_a pool).

Click **Create Virtual Server**.



Create a new Virtual Server

Virtual Server Name: weblogic_example

protocol: HTTP

port: 80

Default Traffic Pool: cluster_a

Create Virtual Server

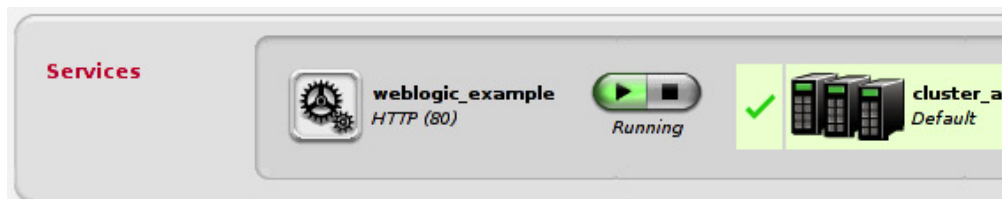
Once the virtual server has been created, you must edit it to make it use the traffic IP group you created earlier. Click on the edit link to the right of the new virtual server's name. Under **Basic Settings**, click the **Traffic IP Groups** radio button, and select the Traffic IP Group you've already created:

The screenshot shows the configuration page for a virtual server named 'weblogic_example'. The 'Internal Protocol' is set to 'HTTP' and the 'Port' is '80'. The 'Default Traffic Pool' is 'cluster_a'. Under 'Listening on:', the 'Traffic IP Groups' radio button is selected. Below this, a table lists the selected Traffic IP Group 'weblogic_example' with a checkmark in the 'Select' column. The 'Domain names and IP addresses' radio button is also visible but not selected. An 'Update' button is at the bottom left.

Traffic IP Group	Select
weblogic_example	<input checked="" type="checkbox"/>
Domain names and IP addresses ...	<input type="checkbox"/>

Starting the virtual server

To start the virtual server, return to the Stingray Traffic Manager Admin Server home page, and click the “play” icon next to your newly created virtual server:



As created, this virtual server will load balance requests across the nodes in your WebLogic cluster using a round robin algorithm. You should be able to test this by visiting one of the traffic IP addresses in your browser. You can use Stingray Traffic Manager's activity graph to confirm that the load balancing is happening correctly.

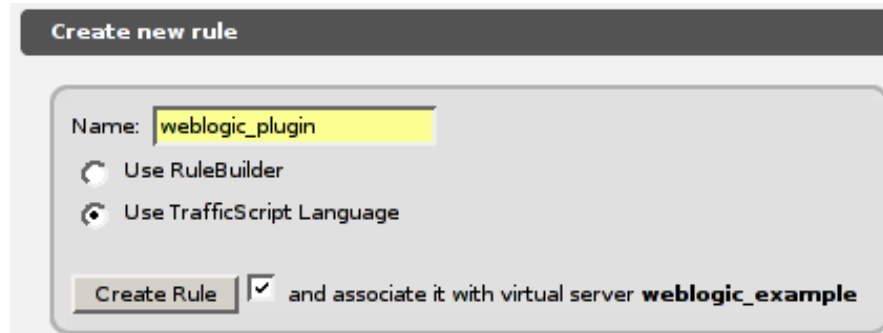
Note, however, that since you haven't yet enabled session persistence, PeopleSoft won't work correctly. Enabling session persistence is discussed later.

Passing the Client IP address to WebLogic Server

Since Stingray Traffic Manager is a proxy, all requests it sends to WebLogic will appear to come from the Stingray Traffic Manager server's internal IP address (10.59.1.20 and 10.59.1.30 in the example). If the application needs to know the true client IP address, Stingray Traffic Manager can pass it to WebLogic in an extra HTTP header. For WebLogic version 6.0 and later, this header must be called "WL-Proxy-Client-IP"; for earlier versions, the header is called "X-Forwarded-For".

First, configure WebLogic to use the extra HTTP header by turning on the **WebLogic Plugin Enabled** setting in the WebLogic administration server. If you are not sure how to do this, please refer to the documentation for your version of WebLogic Server.

Having done this, create a new TrafficScript rule in Stingray Traffic Manager to add the extra header, and enable it in the virtual server for your WebLogic cluster as follows:



Go to **Services > Virtual Servers > Your WebLogic Virtual Server**, and click on the **Rules** link:

- Under **Add new request rule**, click **Manage Rules in Catalog**.
- In the Create new rule dialog, give the rule a name, such as `weblogic_plugin`, select the **"Use TrafficScript language"** radio button, make sure that the "and associate it with ..." tick box is ticked
- Click **Create Rule**.

Cut and paste the rule below that corresponds to the version of WebLogic Server you are using into the rule box, and click Update to create the rule.

WebLogic 6.x, 7.x, 8.x and 9.x:

```
$client_ip = request.remoteIP();  
http.setheader("WL-Proxy-Client-IP", $client_ip);
```

Weblogic 5.x and earlier:

```
$client_ip = request.remoteIP();  
http.setheader("X-Forwarded-For", $client_ip);
```

The rule will take effect immediately, and your Java applications that call `HttpRequest.getRemoteAddr()` will be given the true client IP address.

Enabling Session Persistence

Before PeopleSoft will work correctly, it is necessary to enable session persistence; this causes the second and subsequent requests from a client to be sent to the same back-end server as the first, which prevents server side session state being lost. More recent versions of WebLogic share session state automatically between servers in a cluster; with these versions, enabling session persistence increases the performance of the application by reducing the number of times session data must be replicated between machines in a cluster.

There are three different types of session persistence that are appropriate for use with PeopleSoft:

- **Transparent Session Affinity** With this method, Stingray Traffic Manager simply inserts a cookie into the response header, and uses this cookie to ensure that future requests received from the same client are sent to the same server as the first. This method has the advantage that it is trivial to configure, but the disadvantage that PeopleSoft has no control over the cookie (and so cannot unset it when the user logs out, for example).
- **Monitor Application Cookie** Stingray Traffic Manager can be set to monitor the JSESSIONID cookie that is set by WebLogic. This has the advantage that PeopleSoft retains full control of the cookie.
- **J2EE JSESSIONID cookies/URL (introduced in Stingray Traffic Manager 4.2)** Both of the previous methods rely on the client's browser supporting HTTP cookies. Since not all browsers support cookies, and some users disable cookies as a privacy measure, WebLogic can be configured to add the JSESSIONID to the end of URLs in links generated by the application, when it detects that the user's browser is refusing cookies. Stingray Traffic Manager can extract the JSESSIONID from the requested URL, and use this as its session persistence key to ensure that sessions are persistent regardless of whether the clients support cookies or not.

The Transparent Session Affinity method works simply with PeopleSoft, but all three methods are documented below for reference.

When using the Monitor Application Cookies or Transparent Session Affinity methods, session persistence must be enabled for each of the pools that send traffic to WebLogic. In the example, therefore, it would be enabled for pools "cluster_a" and "cluster_b".

Monitor Application Cookies

Go to **Services > Pools > Your WebLogic Pool** and click on the **Session Persistence** link. Click the **Create New Session Persistence Class** link, and create a class named `weblogic_cookie`:

Create new Session Persistence class

Name:

☒ and associate it with pool **cluster_a**

Set this class to **Monitor Application Cookies** and set the cookie name to `JSESSIONID`:

Monitor application cookies ...
Monitor a specified application cookie to identify sessions.

Cookie:

Leave the failure mode set to **choose a new node to use**. This will cause Stingray Traffic Manager to send the request to a different node if the persistent node isn't available; in this circumstance, the chosen WebLogic server will recover a replica of the session state from a secondary server; the end user should not notice that a failure has occurred.

As soon as you click **Update**, session persistence will be enabled for this pool. You can verify that it is working by looking at the current activity graph as you browse one of the hosted websites. You should see that all the requests in a session are sent to the same backend node. The recent connections page (**Activity > Connections**) can also be used to check that persistence is working correctly.

Transparent Session Affinity

To enable Transparent Session Affinity, create a new session persistence class named "transparent", of type **Transparent Session Affinity**, and set this as your pool's persistence class.

If you haven't created any session persistence classes already, the procedure is similar to that described in Monitor Application Cookie section above.

If you've already created a class, then you must create the new class from the **Catalogs > Persistence** page, and then go to **Services -> Pool -> Your WebLogic Pool > Session Persistence** to set the new class as your pool's session persistence class.

J2EE JSESSIONID cookies/URL

Configuring URL Rewriting Persistence is a simple process; Follow the same procedure as for Transparent Session Affinity, but use **J2EE JSESSIONID cookies/URL** as the persistence class type.

Load Balancing Algorithms

By default, a newly created pool will use a simple round robin algorithm. This takes no account of the load on the back-end servers, and so it is recommended that one of the more sophisticated algorithms is used. The optimal choice depends on the application being run. See the Stingray Traffic Manager User Manual for details of each algorithm².

The perceptive algorithm is a sensible default for a typical PeopleSoft deployment; set it on the **Services > Pool-> Your WebLogic Pool > Load Balancing** page.

SSL Offload

Stingray Traffic Manager can decrypt SSL requests and send the resultant decrypted (plain HTTP) requests to PeopleSoft. This reduces the load on the WebLogic servers, and permits Stingray Traffic Manager's session persistence and advanced content inspection and rewriting features to be used with SSL traffic.

To enable SSL offload, you must first upload your SSL certificate and private key to Stingray Traffic Manager, and then configure SSL decryption for the virtual server concerned. Finally, you must add a TrafficScript rule that will tell WebLogic that the connection to the client was encrypted, so that applications can correctly test whether or not SSL was enabled, and WebLogic can generate correct absolute links (beginning https://) where necessary.

² Stingray Traffic Manager Documentation can be downloaded from <http://riverbed.community.com/docs/>

Uploading your SSL private key and certificate to Stingray Traffic Manager

If you have an existing SSL private key and public certificate, you can upload these files to Stingray Traffic Manager from the **Catalogs > SSL > SSL Certificates Catalog > Import Certificate** page. Enter a name for the certificate pair (the combination of private key and public certificate) and the paths to each of the files:

Import SSL Certificate

This form lets you import an SSL certificate and private key.

Enter a short name to identify your certificate:

Name:

Enter the location of your certificate file:

Certificate file:

Enter the location of your private key file:

Private key file:

If you don't have an existing SSL certificate, you can create a new private key, self-signed certificate and certificate signing request from the **Catalogs > SSL > SSL Certificates Catalog > Create Self Signed Certificate** page. Please refer to the Stingray Traffic Manager User Manual for details.

Configuring SSL Decryption

To enable SSL Decryption, go to the virtual server configuration page for the virtual server concerned (e.g. **Services > Virtual Servers > Your WebLogic Virtual Server**).

First make sure that the port and internal protocol are correct. The internal protocol specifies the protocol after SSL decryption and so should be set to HTTP (the HTTPS protocol is used only when Stingray Traffic Manager is passing unencrypted SSL traffic through to the back-end nodes). The port should be set as required; 443 being the standard HTTPS port. Update the configuration if necessary.

- Click the SSL Decryption link. Set `ssl_decrypt` to yes, and choose an SSL certificate.
- Click the Update button at the bottom of this page.

Your virtual server will now accept and decrypt SSL traffic.

Notifying WebLogic that the connection was encrypted

Add the following rule to the virtual server. This sets the WL-Proxy-SSL header, which tells WebLogic whether or not the original connection between client and Stingray Traffic Manager was encrypted. It's important to explicitly remove this header when the connection isn't encrypted, so that remote clients can't trick WebLogic by adding the header themselves.

```
# Tell WebLogic if the connection to the client was encrypted
# and remove the WL-Proxy-SSL header if not
if ( ssl.isSSL() ) {
    http.setHeader("WL-Proxy-SSL", "true");
} else {
    http.removeheader("WL-Proxy-SSL");
}
```

Running both HTTP and HTTPS versions of a website

If you wish to run both an HTTP (unencrypted) and HTTPS (encrypted) version of your website, simply create two virtual servers that use the same default pool, one set to listen to port 443, with SSL Decryption enabled as described in the preceding section, and the other set to listen to port 80 with SSL decryption disabled. Remember to enable the TrafficScript rules for URL Rewriting Persistence and Client IP Address Passthrough for both virtual servers.

Traffic Routing (Separation of static and dynamic content)

You can use TrafficScript to distinguish requests for static content from those for dynamic content, sending the former to your web servers (.31 and .32 in the example architecture diagram) and the latter to one or more of your application server clusters.

The principle is straightforward - you write a TrafficScript rule that reads details of the request (usually the requested URL or Host header) and uses these to decide which pool to send the requests to. Since the precise details of the rule will vary from application to application, the following is given by way of example only.

To use this rule, you would first create new pools:

- Pool cluster_b contains the WebLogic servers in Application Cluster B (10.59.1.61:80 and 10.59.1.62:80)
- Pool static_servers contains your static web servers (10.59.1.31:80 and 10.59.1.32:80)

Session persistence for cluster_b would be enabled as for cluster_a; session persistence is not normally needed for static content.

Then add the rule below as the final request rule for the "weblogic_example" virtual server:

```
# URLs beginning /public/ are static, everything else is dynamic
$path = http.getPath();

# Send requests for anything other than dynamic content to
# the static cluster
if (! string.regexmatch($path, "^/public/")) {
    pool.use("static_servers");
}

# Split dynamic requests between the two application clusters, according
# to the host header - requests for it.example.com and sales.example.com
# are sent to cluster_a, all other requests to cluster_b.
$host = http.getHeader("Host");
if (string.regexmatch($host, "^(it|sales).example.com$")) {
    pool.use("cluster_a");
} else {
    # Everything else to the second application server cluster
    pool.use("cluster_b");
}
```

About Riverbed

Riverbed delivers performance for the globally connected enterprise. With Riverbed, enterprises can successfully and intelligently implement strategic initiatives such as virtualization, consolidation, cloud computing, and disaster recovery without fear of compromising performance. By giving enterprises the platform they need to understand, optimize and consolidate their IT, Riverbed helps enterprises to build a fast, fluid and dynamic IT architecture that aligns with the business needs of the organization. Additional information about Riverbed (NASDAQ: RVBD) is available at www.riverbed.com.



Riverbed Technology, Inc.
199 Fremont Street
San Francisco, CA 94105
Tel: (415) 247-8800
www.riverbed.com

Riverbed Technology Ltd.
The Jeffreys Building
Cowley Road
Cambridge CB4 0WS
United Kingdom
Tel: +44 (0) 1223 568555

Riverbed Technology Pte. Ltd.
391A Orchard Road #22-06/10
Ngee Ann City Tower A
Singapore 238873
Tel: +65 6508-7400

Riverbed Technology K.K.
Shiba-Koen Plaza Building 9F
3-6-9, Shiba, Minato-ku
Tokyo, Japan 105-0014
Tel: +81 3 5419 1990